MC68000(AC1)

MOTOROLA INC.

MOS Integrated Circuits Division

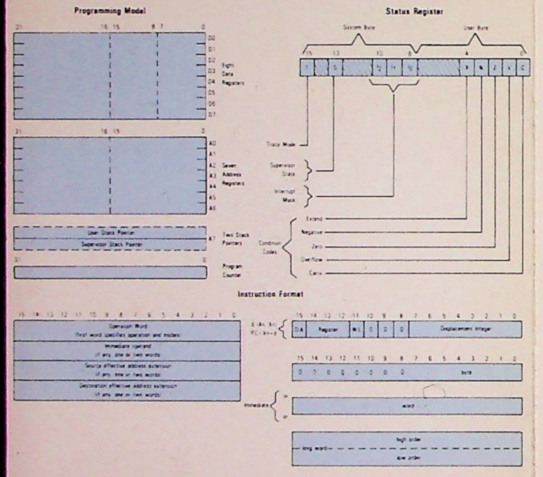
MC68000

16 Bit Microprocessor Programming Card

3501 Ed Bluestein Blvd Austin, Texas 78721 Telephone: (512)928-6000   Issue A

Programming Model

Status Register



Single-Effective-Address-Instruction Operation Word — General Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	X	X	Effective Address Mode			Register		

Effective Addressing Mode Categories

Type	Mode	Register	Generation	Assembler Syntax
Data Register Direct	000	reg. no.	EA = Dn	Dn
Address Register Direct	001	reg. no.	EA = An	An
Register Indirect	010	reg. no.	EA = (An)	(An)
Postincrement Register Indirect	011	reg. no.	EA = (An), An ← An + N	(An)+
Predecrement Register Indirect	100	reg. no.	An ← An − N, EA = (An)	−(An)
Register Indirect With Offset	101	reg. no.	EA = (An) + d16	d(An)
Indexed Register Indirect With Offset	110	reg. no.	EA = (An) + (Xn) + d8	d(An, Xn)
Absolute Short	111	000	EA = (Next Word)	xxx
Absolute Long	111	001	EA = (Next Two Words)	xxxxxx
Relative With Offset	111	010	EA = (PC) + 16	PC relative
Relative With Index and Offset	111	011	EA = (PC) + (Xn) + d8	PC relative + Xn
Immediate	111	100	Data = Next Word(s)	#xxx
Quick Immediate	—	—	Inherent Data	—
Implied Register	—	—	EA = SR, USP, SP, PC	—

Notes:
EA = Effective Address
An = Address Register
Dn = Data Register
Xn = Address or Data Register used as Index Register
SR = Status Register
PC = Program Counter
d8 = Eight bit Offset (displacement)
d16 = Sixteen bit Offset (displacement)
N = 1 for Byte, 2 for Words and 4 for Long Words
( ) = Contents of
← = Replaces

Addressing Mode

Mnemonic Operation	Size	Addr. Mode	Dn #	Dn ~	An #	An ~	(An) #	(An) ~	(An)+ #	(An)+ ~	−(An) #	−(An) ~	d(An) #	d(An) ~	d(An, Xi) #	d(An, Xi) ~	Abs.W #	Abs.W ~	Abs.L #	Abs.L ~
ABCD	B	s = Dn d =	2	6																
Add Digits		s = −(An) d =									2	19								
ADD	B/W	s = Dn d =			ADDA		2	13	2	13	2	15	4	17	4	19	4	17	6	21
Add		d = Dn s =	2	4	2*	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16
Binary	L	s = Dn d =			ADDA		2	22	2	22	2	24	4	26	4	28	4	26	6	30
		d = Dn s =	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22
ADDA	W	d = An s =	2	8	2	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20
Add Address	L	d = An s =	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22
ADDI	B/W	s = Imm d =	4	8	ADDA		4	17	4	17	4	19	6	21	6	23	6	21	8	25
Add Immed	L	s = Imm d =	6	16	ADDA		6	30	6	30	6	32	8	34	8	36	8	34	10	38
ADDQ	B/W	s = Imm3 d =	2	4	2*	4	2	13	2	13	2	15	4	17	4	19	4	17	6	21
Add Quick	L	s = Imm3 d =	2	8	2	8	2	22	2	22	2	24	4	26	4	28	4	26	6	30
ADDX	B/W	s = Dn d =	2	4																
Add Multi		s = −(An) d =									2	19								
precision	L	s = Dn d =	2	8																
		s = −(An) d =									2	32								
AND	B/W	s = Dn d =					2	13	2	13	2	15	4	17	4	19	4	17	6	21
Logical And		d = Dn s =	2	4			2	8	2	8	2	10	4	12	4	14	4	12	6	16
	L	s = Dn d =					2	22	2	22	2	24	4	26	4	28	4	26	6	30
		d = Dn s =	2	8			2	14	2	14	2	16	4	18	4	20	4	18	6	22
ANDI	B/W	s = Imm d =	4	8			4	17	4	17	4	19	6	21	6	23	6	21	8	25
And Immed	L	s = Imm d =	6	16			6	30	6	30	6	32	8	34	8	36	8	34	10	38
ASL, ASR	B/W	count = Dn d =	2	6 + 2n																
Arithmetic		count = #1-8 d =	2	6 + 2n																

*Word only
< Maximum value

Opcode Bit Pattern Key

A	Address Register #	e	Source Effective Address	M	Destination EA Mode	r	Source Register	V	Vector #
C	Test Condition	E	Destination Effective Address	P	Displacement	R	Destination Register		
D	Data Register #	I	Direction; 0 = Right, 1 = Left	Q	Quick Immediate Data	S	Size; 00 = B, 01 = W, 10 = L, 11 = Another Operation		

*Note: for PREDECREMENT mode, register mask d0 d7 a0 a7

*Word only
< Maximum value

Opcode Bit Pattern Key
A Address Register # e Source Effective Address M Destination EA Mode r Source Register V Vector #
C Test Condition E Destination Effective Address P Displacement R Destination Register
D Data Register # I Direction; 0 = Right, 1 = Left Q Quick Immediate Data S Size; 00 = B, 01 = W, 10 = L, 11 = Another Operation

Program Control Instructions

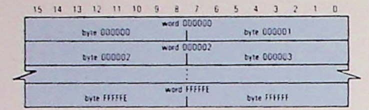
*Word only
< Maximum value

Opcode Bit Pattern Key
A Address Register # e Source Effective Address M Destination EA Mode r Source Register V Vector #
C Test Condition E Destination Effective Address P Displacement R Destination Register
D Data Register # I Direction; 0 = Right, 1 = Left Q Quick Immediate Data S Size; 00 = B, 01 = W, 10 = L, 11 = Another Operation

Conditional Tests

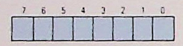
Mnemonic	Condition	Encoding	Test
T	true	0000	1
F	false	0001	0
HI	high	0010	C•Z
LS	low or same	0011	C+Z
CC	carry clear	0100	C
CS	carry set	0101	C
NE	not equal	0110	Z
EQ	equal	0111	Z
VC	overflow clear	1000	V
VS	overflow set	1001	V
PL	plus	1010	N
MI	minus	1011	N
GE	greater or equal	1100	N•V+N•V
LT	less than	1101	N•V+N•V
GT	greater than	1110	N•V•Z+N•V•Z
LE	less or equal	1111	Z+N•V+N•V

Word Organization In Memory

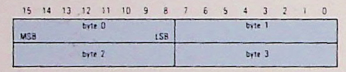


Data Organization In Memory

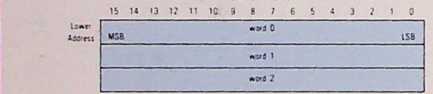
Bit Data
1 byte = 8 bits



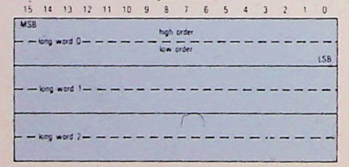
Integer Data
1 byte = 8 bits



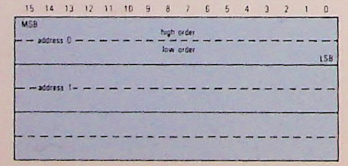
1 word = 16 bits



1 long word = 32 bits

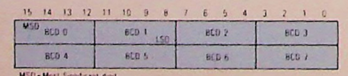


Addresses
1 address = 32 bits



MSB = Most Significant bit
LSB = Least Significant bit

Decimal Data
2 binary coded decimal digits = 1 byte



MSD = Most Significant digit
LSD = Least Significant digit

Exception Vector Assignment

Vector Number(s)	Address Dec	Address Hex	Address Space	Assignment
0	0	000	SP	Reset: Initial SSP
-	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16-23*	64	04C	SD	(Unassigned, reserved)
	95	05F		-
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors
	191	0BF		-
48-63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		-
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		-

*Vector numbers 12, 13, 14, 16 through 23 and 48 through 63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.

Exception Processing Clock Periods

Exception	Periods
Address Error	57
Bus Error	57
Interrupt	47*
Illegal Instruction	37
Privileged Instruction	37
Trace	37

*The interrupt acknowledge bus cycle is assumed to take four external clock periods.

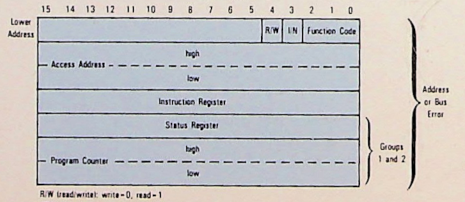
Reference Classification

Function Code Output FC2	FC1	FC0	Reference Class
0	0	0	(Unassigned, Reserved)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned, Reserved)
1	0	0	(Unassigned, Reserved)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

Exception Grouping and Priority

Group	Exception	Processing
0	Reset Bus Error Address Error	Exception processing begins at the next minor cycle
1	Trace Interrupt Illegal Privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV, CHK Zero Divide	Exception processing is started by normal instruction execution

Supervisor Stack Order for Exception



R/W (read/write): write = 0, read = 1
I/N (instruction/not): instruction = 0, not = 1

Exception Vector Format

Word 0	New Program Counter (High)	A0 = 0, A1 = 0
Word 1	New Program Counter (Low)	A0 = 0, A1 = 1

Peripheral Vector Number Format

D15 D8	D7							D0
Ignored	v7	v6	v5	v4	v3	v2	v1	v0

Where
v7 is the MSB of the Vector Number
v0 is the LSB of the Vector Number

Address Translated from 8-Bit Vector Number

A23 A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
All Zeroes	v7	v6	v5	v4	v3	v2	v1	v0	0	0

Operation Code Map

Bits 15 thru 12	Operation
0000	Bit Manipulation/MOVEP/Immediate
0001	Move Byte
0010	Move Long
0011	Move Word
0100	Miscellaneous
0101	ADDQ/SUBQ/Scc/DBcc
0110	Bcc, BSR
0111	MOVEQ
1000	OR/DIV/SBCD
1001	SUB/SUBX
1010	(Unassigned)
1011	CMP/EOR
1100	AND/MUL/ABCD/EXG
1101	ADD/ADDX
1110	Shift/Rotate
1111	(Unassigned)

Dynamic Bit

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			1	Type		Effective Address					

Static Bit

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	Type		Effective Address					

Bit Type Codes: TST = 00, CHG = 01, CLR = 10, SET = 11

MOVEP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			Op Mode			0	0	1	Register		

Op Mode: Word to Reg = 100, Long to Reg = 101, Word to Mem = 110, Long to Mem = 111

OR Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Size		Effective Address					

AND Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	Size		Effective Address					

SUB Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	Size		Effective Address					

ADD Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	Size		Effective Address					

EOR Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	Size		Effective Address					

CMP Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	Size		Effective Address					

MOVE Byte

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	Destination						Source					
				Register			Mode			Mode			Register		

MOVE Long

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	Destination						Source					
				Register			Mode			Mode			Register		

MOVE Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	Destination						Source					
				Register			Mode			Mode			Register		

NEGX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	Size		Effective Address					

MOVE from SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	1	1	Effective Address					

CLR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0	Size		Effective Address					

NEG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	Size		Effective Address					

MOVE to CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	1	1	Effective Address					

NOT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	Size		Effective Address					

MOVE to SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	1	1	Effective Address					

NBCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	0	Effective Address					

PEA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	Effective Address					

SWAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	0	0	0	Register		

MOVEM Registers to EA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	1	Sz	Effective Address					

Sz: Long = 1, Word = 0

EXTW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	1	0	0	0	0	Register		

EXTL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	1	1	0	0	0	Register		

TST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	Size		Effective Address					

TAS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	1	1	Effective Address					

MOVEM EA to Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	1	Sz	Effective Address					

Sz: Long = 1, Word = 0

TRAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	0	Vector			

LINK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	0	Register		

UNLK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	1	Register		

MOVE to USP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	0	Register		

MOVE from USP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	1	Register		

RESET

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0

NOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1

STOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	0

RTE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1

RTS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	1

TRAPV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	0

RTR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	1

JSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	0	Effective Address					

JMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	1	Effective Address					

CHK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	Register			1	1	0	Effective Address					

LEA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	Register			1	1	1	Effective Address					

ADDQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Data			0	Size		Effective Address					

SUBQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Data			1	Size		Effective Address					

Scc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Condition				1	1	Effective Address					

DBcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Condition				1	1	0	0	1	Register		

Bcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	Condition				8 bit Displacement							

BSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1	8 bit Displacement							

MOVEQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	Register			0	Data							

OR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Register			Op Mode			Effective Address					

Op Mode

.B	.W	.L	
000	001	010	Dn + EA → Dn
100	101	110	EA + Dn → EA

DIVU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Register			0	1	1	Effective Address					

DIVS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Register			1	1	1	Effective Address					

SBCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Destination Register			1	0	0	0	0	R/M	Source Register		

R/M (register/memory): register-register = 0, memory-memory = 1

SUB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	Register			Op Mode			Effective Address					

Op Mode

.B	.W	.L	
000	001	010	Dn - EA → Dn
100	101	110	EA - Dn → EA
	011	111	An - EA → An

SUBX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	Destination Register			1	Size		0	0	R/M	Source Register		

R/M (register/memory): register-register = 0, memory-memory = 1

CMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Register			Op Mode			Effective Address					

Op Mode

.B	.W	.L	
000	001	010	Dn-EA
	011	111	An-EA

CMPM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Register			1	Size		0	0	1	Register		

EOR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Register			1	Size		Effective Address					

AND

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Register			Op Mode			Effective Address					

Op Mode

.B	.W	.L	
000	001	010	Dn•EA → Dn
100	101	110	EA•Dn → EA

MULU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Register			0	1	1		Effective Address				

MULS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Register			1	1	1		Effective Address				

ABCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Destination Register			1	0	0	0	0	R/M	Source Register		

R/M (register/memory): register-register = 0, memory-memory = 1

EXGD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Data Register			1	0	1	0	0	0	Data Register		

EXGA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Address Register			1	0	1	0	0	1	Address Register		

EXGM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Data Register			1	1	0	0	0	1	Address Register		

ADD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Register			Op Mode			Effective Address					

Op Mode

.B	.W	.L	
000	001	010	Dn + EA → Dn
100	101	110	EA + Dn → EA
	011	111	An + EA → An

ADDX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Destination Register			1	Size		0	0	R/M	Source Register		

R/M (register/memory): register-register = 0, memory-memory = 1

Data Register Shifts

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	Count/Register			d	Size		i/r	Type		Register		

Memory Shifts

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	Type		d	1	1	Effective Address					

Shift Type Codes: AS = 00, LS = 01, ROX = 10, RO = 11
d (direction): Right = 0, Left = 1
i/r (count source): Immediate Count = 0, Register Count = 1

Condition Code Computations

Operations	X	N	Z	V	C	Special Definition
ABCD	*	U	?	U	?	C = Decimal Carry $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
ADD, ADDI, ADDQ	*	*	*	?	?	$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$
ADDX	*	*	?	?	?	$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	–	*	*	0	0	
CHK	–	*	U	U	U	
SUB, SUBI SUBQ	*	*	*	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
SUBX	*	*	?	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
CMP, CMPI, CMPM	–	*	*	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
DIVS, DIVU	–	*	*	?	0	V = Division Overflow
MULS, MULU	–	*	*	0	0	
SBCD, NBCD	*	U	?	U	?	C = Decimal Borrow $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
NEG	*	*	*	?	?	$V = Dm \cdot Rm$, $C = Dm + Rm$
NEGX	*	*	?	?	?	$V = Dm \cdot Rm$, $C = Dm + Rm$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
BTST, BCHG, BSET, BCLR	–	–	?	–	–	$Z = \overline{Dn}$
ASL	*	*	*	?	?	$V = Dm \cdot (\overline{D_{m-1}} + \ldots + \overline{D_{m-r}}) + \overline{Dm} \cdot (D_{m-1} + \ldots + D_{m-r})$ $C = D_{m-r+1}$
ASL (r = 0)	–	*	*	0	0	
LSL, ROXL	*	*	*	0	?	$C = D_{m-r+1}$
LSR (r = 0)	–	*	*	0	0	
ROXL (r = 0)	–	*	*	0	?	C = X
ROL	–	*	*	0	?	$C = D_{m-r+1}$
ROL (r = 0)	–	*	*	0	0	
ASR, LSR, ROXR	*	*	*	0	?	$C = D_{r-1}$
ASR, LSR (r = 0)	–	*	*	0	0	
ROXR (r = 0)	–	*	*	0	?	C = X
ROR	–	*	*	0	?	$C = D_{r-1}$
ROR (r = 0)	–	*	*	0	0	

– Not affected
U Undefined
? Other – see Special Definition
*General Case:
X = C
N = Rm
$Z = \overline{Rm} \cdot \ldots \cdot \overline{R0}$
Sm – Source operand most significant bit
Dm – Destination operand most significant bit
Rm – Result bit most significant bit
n – bit number
r – shift amount

Powers Of 2, Powers Of 16

16^m m =	2^n n =	Value
0	0	1
	1	2
	2	4
	3	8
1	4	16
	5	32
	6	64
	7	128
2	8	256
	9	512
	10	1,024
	11	2,048
3	12	4,096
	13	8,192
	14	16,384
	15	32,768
4	16	65,536
	17	131,072
	18	262,144
	19	524,288
5	20	1,048,576
	21	2,097,152
	22	4,194,304
	23	8,388,608
6	24	16,777,216
	25	33,554,432
	26	67,108,864
	27	134,217,728
7	28	268,435,456
	29	536,870,912
	30	1,073,741,824
	31	2,147,483,648
8	32	4,294,967,296

ASCII Character Set (7 Bit Code)

LS Dig. \ MS Dig.	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Hexadecimal and Decimal Conversion

How to use:

Conversion to Decimal: Find the decimal weights for corresponding hexadecimal characters beginning with the least significant character. The sum of the decimal weights is the decimal value of the hexadecimal number.

Conversion to Hexadecimal: Find the highest decimal value in the table which is lower than or equal to the decimal number to be converted. The corresponding hexadecimal character is the most significant. Subtract the decimal value found from the decimal number to be converted. With the difference repeat the process to find subsequent hexadecimal characters.

23 Byte 16				15 Byte 8				7 Byte 0			
23 Char 20		19 Char 16		15 Char 12		11 Char 8		7 Char 4		3 Char 0	
Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3840	F	240	F	15

A11305 PRINTED IN USA 9/80 IMPERIAL LITHO 689411 30M